Complexity of Path Covering Problems in Acyclic Alternate Graphs II(Algorithms : Mathematical Foundations and Applications)

AUTHOR(S):

Uemura, Kenji; Yaku, Takeo

数理解析研究所講究録
591 巻 1986 年 12-24

Complexity of Path Covering Problems in Acyclic Alternate Graphs II

Kenji Uemura
植村憲治
Department of Mathematics
Tsuru University

Takeo Yaku
夜久竹夫
Department of Mathematical Science
Tokyo Denki University

ABSTRACT

A path cover [3,1978] of a digraph is a partition of digraph into node disjoint paths. It is also called a linearlization [1,1975].

In this paper we first show a k x m running time algorithm (k is a constant, m is the number of edges) that obtains a path cover with the minimum number of paths for directed acyclic graphs (dags) with either the outdegrees or the indegrees bounded by two. The constant k is less than 1.5 in expected case and is 2 in worst case. Using this algorithm, we consider a bound for the average running time of path-covering problems of acyclic alternate graphs. Finally we note that the numerical evaluations give the conjecture that the complexity is bounded by n + log n.

1. INTRODUCTION.

There have been many studies about covering of graphs by Yaku-Iwata (see Iwata[3,1978]), Boesch-Gimpel [2, 1977], Iwata [3,1978] and Yaku[5,1979], Boffey[1,1975] and Ramanath-Solomon [4,1983]. This paper concerns to problems to minimumize the number of paths in such covering. We note that a path covering with the mininum path number is called maximal, since the number

of the edges covered by a covering is maximal when the path number is minimum.

Finding problem of a path cover with the minimum number of paths is NP-complete for a digraph. Algorithms are known that run in $o(m^{2.5})$ time for dags , and that runs in $o(m)$ time for dags with both the outdegrees and the indegrees bounded by two, where m is the number of the edges.

We deal with the class of dags with the outdegrees bounded by two which concerns to data structures with the branches bounded by two such as program flowcharts with binary branches.

We relate the approximate running time of the algorithm given in [5] within the case of acyclic alternate graphs of finding a maximal path cover.

Concerned with the approximate time complexity, we calculate recursively defined functions which increase less than A^{n^t} for any $t > 1$, but more than A^n.

2. Preliminary definitions.

Definition. Let G=(V,E) be a digraph. Edges (u,v) and (w,x) in E are alternately adjacent, denoted by $(u,v) \sim (w,x)$, if they are distinct and u=w or v=x.

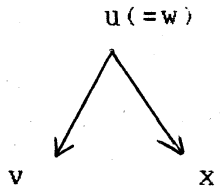

Fig.1 (a)

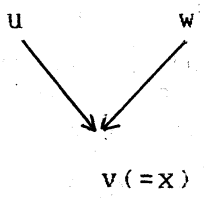

Fig.1 (b)

The symbol $\overset{*}{\sim}$ denotes the reflexive and transitive closure of $\sim$. Edges e and e' are alternately equivalent if $e \overset{*}{\sim} e'$. A graph G=(V,E) is alternate if $e \overset{*}{\sim} e'$ for any edges e and e'.

We assume that our alternate graph satisfies next two conditions.

(1) The indegree of any node is no more than 3 and the outdegree of any node is no more than 2.

(2) Graphs are acyclic even if we neglect the direction of arrow.

We also assume that there is one special node of outdegree $\neq 0$ called a root and the graph is called a rooted altetnate graph.

We will evaluate the approximate running time, assuming each non-isomorphic rooted alternate graph to appear under the same probability.

Let $G_1=(V_1,E_1)$, $G_2=(V_2,E_2),\ldots,G_n=(V_n,E_n)$ be subgraphs of a digraph G=(V,E). If $V_1 \cup V_2 \cup \ldots \cup V_n = V$ and $V_i \cap V_j = \phi\ (i \neq j)$, then $P=\{G_1,G_2,\ldots,G_n\}$ is a partition of G. Each subgraph is called a component(block). The partition P is a path cover if each $G_i\ (i \leq n)$ satisfies either

(i) G_i is connected and there is a vertex disjoint path $e_1e_2\ldots e_l\ (l>1)$ in G_i such that $\{e_1,e_2,\ldots e_l\}=E_i$, or

(ii) G_i is a point, that is, $G_i=(\{v\},\phi)\ (v \in V)$.

For a path cover $P=\{G_1,G_2,\ldots,G_n\}$ of a graph G=(V,E), G(P) denotes the graph $G(P)= G_1 \cup G_2 \cup \ldots \cup G_n$.

The path cover P is maximal, if for any path cover Q, $\#(E-E(P))<\#(E-E(Q))$, that is $\#E(P)>\#E(Q)$.

3. Algorithms of finding a maximal path cover.

```
    procedure ALTSEARCH(v,v_s,flag,G,H)
/* comment  alternate trace with backward modification          */
/* data  G : input; an alternate dag, initially the             */
/*           vertices in G are marked "UNCOVERED" or            */
/*           "COVERED", and  "UNVISITED" or "VISITED".          */
/*       v : input; a vertex in G, the current trace starts     */
/*           from v.                                            */
/*       v_s : input; the trace originally started from v_s     */
/*      flag : input; the indicator whether the output tree     */
/*                    is already modified backward, initially   */
/*                    flag is marked "UNMODIFIED".              */
/*       H : output;  a spanning out tree of G                  */
     do  mark v "VISITED" ;
     iterate while vertex u remaining in OUTLIST(v, G)
        marked "UNCOVERED"  do
        add (v,u) to H ; mark u "COVERED" ;
        iterate while vertex w remaining in INLIST(u,G)
        marked "UNVISITED"   do
        if (there is no vertex x (x≠u) in OUTLIST(w,G) .or.
        there  is  the  vertex x(x≠u) in  OUTLIST(w,G)  marked
        "COVERED") .and. flag is marked "UNMODIFIED" then do
             call ALTMODIFY(w,v_s,P,G,H) ;
             mark flag "MODIFIED"   od ;
        call ALTSEARCH(w, v_s,flag, G, H) od  od  od

procedure ALTMODIFY(v_e, v_s, P, G, H)
/* comment   subprocedure that modifies a spanning out forest   */
```

```
/*          H in  ALTSEARCH                                          */
/* data     v_s, G ,H are the same in ALTSEARCH                      */
/*          v_e: input ; a vertex in G, a modificatio.       of      */
/*                   ALTMODIFY starts from "encountered" vertex     */
/*                   v_e to "starting" vertex v_s.                  */
/*          P : work ; the semipath from v_s to v_e in the trace    */
/*                   of ALTSEARCH.                                  */
     do   v:=v_e;
     let P be the alternate semipath from v_s to v_e in the current
          edge trace by ALTSEARCH(v ,v_s ,flag , G, H)
     call ALTBACKWARD(v_e , v_s, P ,G ,H)   od ;

procedure ALTBACKWARD(v , v_s ,P ,G ,H)
/* comment  subprocedure of ALTMODIFY                               */
/* data  v : input ; backward modification currently starts         */
/*                   from v                                         */
/*       v_s : input ; terminal vertex of backward modification,    */
/*                which is the starting vertex of ALTSEARCH         */
/*       P : input/work/output ; the alternate semipath from        */
/*            v_s to v in the trace of ALTSEARCH                    */
/*       G : input ; the "input" graph                              */
/*       H : input / output ; the spanning out forest to be         */
/*            modified                                              */
     do  find the edge e in P of the form (v, u) ;
     add (v, u) to H ;
     find the edge in P of the form (w, u) (v≠w) ;
     delete (w, u) from H ;
     /* initially (w, u) is in H
```

call while $w \neq v_s$

ALTBACKWARD(w, v_s, P-(w, u)(v, u), G, H);

od;

Outline of these algorithms.

From the starting vertex, do the depth first search (dfs), marking going away edges till having reached a terminal vertex of outdegree=1 or a loop completing vertex of outdegree=0. If it is the first time of the flag being "UNMODIFIED", then come back on the semipath to the originally starting vertex, deleting going away edges and marking coming back edges and continue dfs marking going away edges. If it is not the first time then continue dfs.

Theorem 1. Algorithm ALTSEARCH obtains a maximal path cover of a dag with unbounded indegrees and the outdegrees bounded by two in 2 x m worst time and 1.5 x m average time.

4. Mean length of alternate graphs.

Definition. The mean length of returning semipaths of any rooted alternate graph G is defined by

$\sum_{a:\text{leaf of G of outdegree=1}} p(a)*q(a)$: $p(a)=1/2^k$, and k is the number of branches on the semipath from root to a, such that each subalternate graph having which as the root has at least one leaf of outdegree=1 ; q(a) is the length of this semipath.

The mean length of returning paths of n-node alternate graphs is defined by SML(n)/NAL(n), where NAL(n)is the number of all n-node rooted alternate graphs satisfying conditions (1) and

(2), and SML(n) is the total sum of mean lengths of returning paths of these n-node alternate graphs.

Proposition 1. Let NALT(n,k) be the number of n-node alternate graphs whose roots are terminal and the number of leaves of outdegree=1 are k. And SMLT(n,k) be the total sum of mean lengths of returning paths of these n-node alternate graphs. NALT(n) be the summation of all NALT(n,k), and SMLT(n) be the summation of all SMLT(n,k). Then next equations hold.

$$(3)\quad NALT(n,k)=\Big(\sum_{\substack{j+l=n-2\\ 0\leqq j<l}}\ \sum_{\substack{i+h=k\\ 0\leqq i\leqq k}} + \sum_{\substack{j=l,\ i+h=k\\ 0\leqq i<h}}\Big)\ NALT(j,i)NALT(l,h)$$

$$+NALT(n/2-1,k/2)\{NALT(n/2-1,k/2)+1\}/2$$

$$NALT(0,0)=1, NALT(1,0)=0, NALT(1,1)=NALT(2,0)=1,$$

$$NALT(2,1)=0.$$

$$(4)\quad NAL(n,k)=\Big(\sum_{\substack{j+l=n+1\\ 2\leqq j<l}}\ \sum_{\substack{i+h=k\\ 0\leqq i\leqq k}} + \sum_{\substack{j=l,\ i+h=k\\ 0\leqq i<h}}\Big)NALT(j,i)NALT(l,h)$$

$$+NALT((n+1)/2,k/2)\{NALT((n+1)/2,k/2)+1\}/2$$

$$(5)\quad SMLT(n,k)=2\times NALT(n,k)+\sum_{\substack{j+l=n-2\\ 1\leqq j\leqq n-2}} SMLT(j,k)NALT(l,0)$$

$$+\Big(\sum_{\substack{j+l=n-2\\ 1\leqq j<n-2}}\ \sum_{\substack{i+h=k\\ 1\leqq i<k}} SMLT(j,i)NALT(l,h)$$

$$+SMLT(n/2-1,k/2)\Big)/2$$

$$SMLT(0,0)=SMLT(1,0)=0$$

$$(6)\quad SML(n,k)=SMLT(n,k)+\sum_{\substack{j+l=n+1\\ 1<j\leqq n}} SMLT(j,k)NALT(l,0)$$

$$+\Big(\sum_{\substack{j+l=n+1\\ 2\leqq j\leqq n-1}}\ \sum_{\substack{i+h=k\\ 0<i<k}} SMLT(j,i)NALT(l,h)$$

$$+SMLT((n+1)/2,K/2)\}/2$$

If we add one more condition of the outdegree of every terminal node being one, and limit our discussin to this class of alternate graphs, then equations of NALT(n), NAL(n), SMLT(n), SML(n) become simpler. And in this case we obtain the proof of these four equations increasing larger than A^n but less than A^n for any $t > 1$. This may give us some knowledge about upper bounds of SML(n)/NAL(n).

Proposition 2. Let NALT(n) be the number of n-node alternate graphs whose roots are terminal nodes and SMLT(n) be the total sum of mean length of these n-node alternate graphs. Then next equations hold.

$$(7)\quad NALT(2n)=\sum_{i=0}^{n-2}\{NALT(2n-2-i)NALT(i)\}+NALT(n-1)\{NALT(n-1)+1\}/2 \qquad (n>2)$$

$$NALT(2n+1)=\sum_{i=0}^{n-1}NALT(2n-1-i)NALT(i) \qquad (n>1)$$

$$NALT(0)=NALT(1)=NALT(2)=1$$

$$(8)\quad NAL(2n)=\sum_{i=1}^{n}NALT(2n+1-i)NALT(i)$$

$$NAL(2n+1)=\sum_{i=1}^{n}NALT(2n+2-i)NALT(i)+NALT(n+1)\{NALT(n+1)+1\}/2$$

$$(9)\quad SMLT(2n)=SMLT(2n-2)+\{\sum_{i=1}^{2n-3}SMLT(2n-2-i)NALT(i)+SMLT(n-1)\}/2$$

$$+2*NALT(2n) \qquad (n \geq 1)$$

$$SMLT(2n+1)=SMLT(2n-1)+\sum_{i=1}^{2n-2}SMLT(2n-1-i)NALT(i)/2$$

$$+2*NALT(2n+1) \qquad (n \geq 1)$$

$$SMLT(0)=SMLT(1)=0$$

$$(10)\quad SML(2n)=SMLT(2n)+\{\sum_{i=2}^{2n-1} SMLT(2n+1-i)NALT(i)\}/2$$

$$SML(2n+1)=SMLT(2n+1)+\{\sum_{i=2}^{2n} SMLT(2n+2-i)NALT(i)+SMLT(n+1)\}/2$$

5. Evaluations of previous sequences (7)-(10).

Lemma. For any $t>1$ and $A>1$, there exists some N such that for any $n>N$, following three conditions hold,

a) $\log_A n < n$

b) $(6n+1)^t > (6n-1)^t+1$

c) $((3n+i)^t+(3n-i-1)^t)-((3n+i-1)^t+(3n-i)^t) > 0 \qquad (i>0)$

$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad > 1 \quad (3n>i>n).$

proof. a) Since for any $n>N$ $n<A^n$,

then $\log_A n < n$.

b) Trivial.

c) Consider the next function,

$$f(x)=(3n+x)^t+(3n-x-1)^t \qquad 0< x < 3n-1.$$

Then $f'(x)=t\{(3n+x)^{t-1}-(3n-x-1)^{t-1}\} > 0.$

And $f(n)=t\{4n^{t-1}-(2n-1)^{t-1}\} > t*n^{t-1}(4^{t-1}-2^{t-1}) > 1$ for any $n>N$.

Therefore $(3n+i)^t+(3n-i-1)^t-\{(3n+i-1)^t+(3n-i)^t\}$

$= f(i) - f(i-1) > 1 \quad (i > n)$.

Theorem 2. Previous sequences defined by equations (7)-(10) increase larger than A^n but less than A^{n^t} for any $t > 1$.

proof. Comparing with Fibonacci sequence, we can verify easily that these four sequences increase as large as A^n at least.

i) First we prove $NALT(n) > O(A^n)$.

Assume that $NALT(n) = O(A^n)$ for some $A>0$. Then there is some $c>0$, and for any $\varepsilon > 0$, there exists a number N such that for any $n > N$, $c - \varepsilon < NALT(n)/A^n < c + \varepsilon$.

$$NALT(4n+1) = NALT(4n-1)+ NALT(4n-2)+...+NALT(3n-1)NALT(n)+...$$
$$+NALT(2n)NALT(2n-1)$$
$$> NALT(3n-1)NALT(n)+...+NALT(2n)NALT(2n-1)$$
$$>(n-1)A^{4n-1}(c-\varepsilon)^2.$$

Then $NALT(4n+1)/A^{4n+1}=(n-1)(c-\varepsilon)^2/A^2 > c+\varepsilon$.

This is a contradiction. So $NALT(n) > O(A^n)$.

The rest three sequences have the same properties since they increase larger than NALT(n).

ii) Let $NALT(n)=O(A^{n^t})$ for some $t>1$. Then for any > 0 there exists some N such that for any $n > N$,

$NALT(n)/A^{n^t} < c+\varepsilon$,and for some $n > N$, $c-\varepsilon < NALT(n)/A^{n^t}$.

Let $M=\max\{NALT(i)/A^{i^t} \; i=0,1,...,N\}$. Then $NALT(i) < M*A^{i^t}$.

$$NALT(6n+1)=NALT(6n-1)+NALT(6n-2)+...+NALT(5n-1)NALT(n)+...$$
$$+NALT(3n)NALT(3n-1)$$
$$<(c+\varepsilon)M(A^{(6n-1)^t+0}+...+A^{(5n-1)^t+n^t})+(c+\varepsilon)^2(A^{(5n-2)^t+(n+1)^t}$$
$$+...+A^{(4n-1)^t+(2n)^t}+A^{(4n-2)^t+(2n+1)^t}+...+A^{(3n)^t+(3n-1)^t}).$$

Since $A^{(4n-2)^t+(2n+1)^t}+...+A^{(3n)^t+(3n-1)^t}$

$$<(n-1)A^{(4n-2)^t+(2n+1)^t}$$

$$= A^{\log_A(n-1)} * A^{(4n-2)^t + (2n+1)^t}$$

$$< A^{n-1} A^{(4n-2)^t + (2n+1)^t},$$

$$NALT(6n+1) < A^{(6n-1)^t + 0^t + 1} \quad \text{for any } n > N.$$

Then $NALT(n) \neq O(A^{n^t})$, and it is easily shown that

$NALT(n) < O(A^{n^t})$.

Next, $NAL(2n) = \sum_{i=1}^{n} NALT(2n+1-i)NALT(i) < \sum_{i=0}^{n} NALT(2n+1-i)NALT(i)$

$$= NALT(2n+3)$$

Then $NAL(2n) < NALT(2n+3)$. Similarly $NAL(2n+1) < NALT(2n+4)$.

Then $NAL(n) = O(NALT(n)) < O(A^{n^t})$.

The same argument shows that $SMLT(n) < O(A^{n^t})$ and $SML(n) < O(A^{n^t})$ for any $t > 1$.

This theorem sugests that SML(n)/NAL(n) increases slowly, or less than $O(A^{n^t - n})$ for any $t > 0$.

6. Numerical evaluation.

According to numerical examples below, SML(n)/NAL(n) (equations (6),(3)) seems to increase less than O(log n).

n;	NAL(n);	SML(n);	SML(n)/NAL(n);	(SML(n)/NAL(n))/log n;
20	15229	89992.0	5.90925	1.97255
40	1.64596E+09	1.29418E+10	7.86272	2.13146
60	2.46558E+14	2.13401E+15	8.65520	2.11394

80	4.28494E+19	3.89030E+20	9.07898	2.07186
100	8.10934E+24	7.57691E+25	9.34345	2.02890
120	1.62203E+30	1.54489E+31	9.52439	1.98943

These evaluations give the conjecture that the approximate running time of algorithm ALTMATCH on alternate graph of node n satisfying conditions (1) and (2) is bounded by

n+log n .

7. Concluding Remarks.

Our next aims are at first deleting the condition of terminal node and extending to the class of 2-n bounded graphs . Then deleting acyclicness. The method we take in the evaluation of order of recursively defined functions may be also useful to determine the order of some other those functions, and we will be able to get some conditions of recursively defined functions which tell us the bound of increasing.

References.

1. T. B. Boffey, The linearization of flowcharts, BIT 15 (1975), 141 - 150.

2. F. T. Boesch and J. F. Gimpel, Covering the points of a digraph with point-disjoint paths and its application to code optimization, J. ACM 24 (1977), 192 - 198.

3. S. Iwata, Programs with minimal goto statements, Inform. Contr. 37 (1978) .

4. M. Ramanath and M. Solomon, Jump minimumiation in linear time, ACM Transac. Programming Languages and Systems 6 (1984).

5. T. Yaku, A linear time algorithm that obtains a maximal matching for graphs, Memoire of the RIMS Koto Univ. 353 (1979), (in Japanese)

6. T. Yaku, Partition of graphs into chains of the minimum block number, Proc. Facul. Sci. Tokai Univ. 18 (1983), 41 - 44.

7. K. Uemura, T. Yaku, Complexity of Path covering problems in Acyclic Alternate Graphs, Memoire of the RIMS Kyoto Univ. 556 (1985), 240-249.